ECE 696B: Spring 2025
Trustworthy Machine Learning

Lecture 13A: *Extracting training data from LLMs*

Instructor: Dr Ravi Tandon

Department of ECE

Lecture Outline

- Published in 2021 USENIX Security
- Cited **2025 times**

  (as of March 3, 2025)
- Showed vulnerability of LLMs
- Training data can be extracted
- Principled approach for extraction

Extracting Training Data from Large Language Models

Nicholas Carlini[1] Florian Tramèr[2] Eric Wallace[3] Matthew Jagielski[4]

Ariel Herbert-Voss[5,6] Katherine Lee[1] Adam Roberts[1] Tom Brown[5]

Dawn Song[3] Úlfar Erlingsson[7] Alina Oprea[4] Colin Raffel[1]

[1]*Google* [2]*Stanford* [3]*UC Berkeley* [4]*Northeastern University* [5]*OpenAI* [6]*Harvard* [7]*Apple*

THE UNIVERSITY OF ARIZONA

Conventional wisdom

- Privacy leakage typically associated with ***Overfitting***
- *(informal)* **Definition of Overfitting:** training loss << test loss
- LLMs do not quite exhibit overfitting: **both test & training losses low**
- LLMs seem to work very well on unseen (zero-shot) problems
- LLMs often trained on de-duplicated data over 1 epoch
- One epoch = Each data point is only "seen" once during training
- → ***Conventional wisdom = LLMs should not significantly memorize training data***

Contributions of this paper

- LLMs are prone to memorizing training data
- *While LLMs may not memorize on average, there are some "worst-case" examples that are memorized*
- Proposes principled approach for extracting training data
- In-depth study on GPT-2 (large enough model for that time..2021)
- Focus on GPT-2 XL (1.5 Billion parameters)

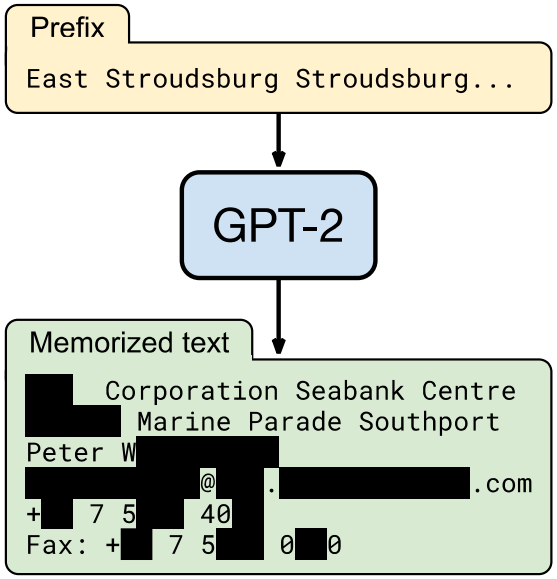

Figure 1: **Our extraction attack.** Given query access to a neural network language model, we extract an individual person's name, email address, phone number, fax number, and physical address. The example in this figure shows information that is all accurate so we redact it to protect privacy.

THE UNIVERSITY OF ARIZONA

Towards Defining **Language Model Memorization**

- Some memorization in LLMs is *essential* (for them to work)..
- We care about defining *Unintended* memorization
- Authors borrow inspiration from the concept of Eidetic Memory
- Definition of Eidetic or Photographic memory: ability to recall information after seeing it only once.

Eidetic Memorization of Text

Definition 1 (Model Knowledge Extraction) *A string s is* extractable[4] *from an LM f_θ if there exists a prefix c such that:*

$$s \leftarrow \underset{s' :\ |s'|=N}{\arg\max}\ f_\theta(s' \mid c)$$

- f(s'|c) = likelihood of model output being s' given c as an input
- Arg max can be replaced by other practical sampling strategy

Definition 2 (k-Eidetic Memorization) *A string s is k-eidetic memorized (for $k \geq 1$) by an LM f_θ if s is extractable from f_θ and s appears in at most k examples in the training data X: $|\{x \in X : s \subseteq x\}| \leq k$.*

- What is an Example: for GPT-2, a webpage (in entirety) is an example
- A string can appear many times in that "example" so will still treat k=1
- Ideally one would like as many strings to be extractable for as small k as possible!

Threat Model & Attacks

- Attacker only has black-box access to the LLM
- Paper focuses on GPT-2 family (primarily XL; 1.5B parameters)
- Data sources used for training GPT-2 known publicly (declared by OpenAI); which data points itself were used is not known.
- Attacks are done to **indiscriminately** extract training data

(Authors *did not* try targeted extraction of training data)

- Ethical considerations: authors reported their findings with OpenAI

Two-Step Attack Procedure

- **Step 1: Text Generation from LLM**
- **Step 2: Predict which outputs contain memorized text**

Baseline Approach

- **Text generation**: give a start token and generate 256 tokens using top-n sampling.
- Measure Perplexity: $\mathcal{P} = \exp\left(-\frac{1}{n}\sum_{i=1}^{n}\log f_\theta(x_i|x_1,\ldots,x_{i-1})\right)$
- Low perplexity = High likelihood = Predict sample was seen in training
- This approach works and reveals/extracts..
  - MIT public license (seen in many strings)
  - User guidelines of Vaughn Live (an online streaming site)
  - Showed Twitter handles or e-mail addresses of people
- **Issues with this approach:**
  - Low diversity in outputs (issue in generation)
  - Perplexity based MIA has too many false positives (issue in detection)

Step-1 improvements (better generation)

- Sampling with decaying temperature (scaling softmax)
- Z= logits; Instead of Prob = Softmax(Z); Prob' = Softmax(Z/T) for some T>1
- T = "temperature"; larger T = less confident and more diverse outputs
- Improvement 1: Authors propose a decaying temperature schedule
  - (start with t=10, …reduce..t=1) over 20 tokens;
  - explore in the beginning of the generation and then
  - follow high confidence path/completion.
- Improvement 2: Better conditioning (use starting text from CommonCrawl)

Step-2 improvements (better prediction/MIA)

- Several types of strings are assigned high likelihood (low Perplexity)
  - "Trivial Memorization"; e.g., numbers 1-100
  - Repeated sub-strings; e.g., "I love you"

- **Idea 1: Use Another LLM as a Filter**
  - Train it (say LLM-2) using some other dataset
  - If LLM-1 and LLM-2 *both* assign high probability, then this can be filtered..
- **Idea 2: Compress the string & measure "surprise" (zLib compressor)**
  - Compress the string using zlib
  - Measure "entropy"; larger entropy = "surprising"
  - Authors use Ratio of Perplexity/Entropy as the Membership inference statistics
- **Idea 3: Compare to lowercase text (same string in Upper vs. Lowercase)**
- **Idea 4: Perplexity (averaged) over a sliding window over the string**

Summary of 2-Step Attack Strategy

- *Top-n* (§4.1) samples naively from the empty sequence.
- *Temperature* (§5.1.1) increases diversity during sampling.
- *Internet* (§5.1.2) conditions the LM on Internet text.

We order each of these three datasets according to each of our six membership inference metrics:

- *Perplexity*: the perplexity of the largest GPT-2 model.
- *Small*: the ratio of log-perplexities of the largest GPT-2 model and the Small GPT-2 model.
- *Medium*: the ratio as above, but for the Medium GPT-2.
- *zlib*: the ratio of the (log) of the GPT-2 perplexity and the zlib entropy (as computed by compressing the text).
- *Lowercase*: the ratio of perplexities of the GPT-2 model on the original sample and on the lowercased sample.
- *Window*: the minimum perplexity of the largest GPT-2 model across any sliding window of 50 tokens.

Overall Framework for Data Extraction

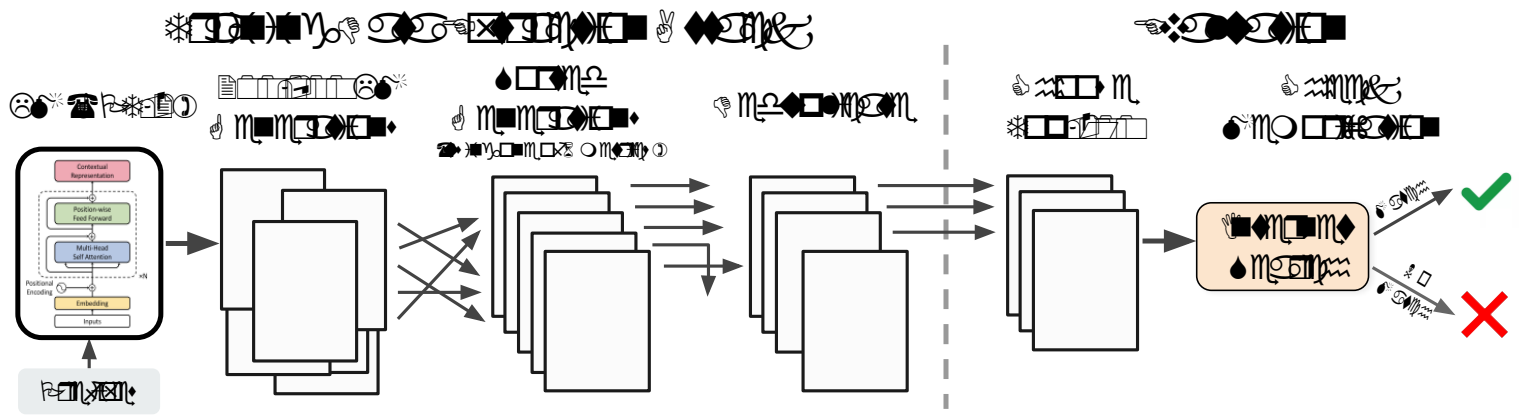

Figure 2: Workflow of our extraction attack and evaluation. 1) Attack. We begin by generating many samples from GPT-2 when the model is conditioned on (potentially empty) prefixes. We then sort each generation according to one of six metrics and remove the duplicates. This gives us a set of potentially memorized training examples. 2) Evaluation. We manually inspect 100 of the top-1000 generations for each metric. We mark each generation as either memorized or not-memorized by manually searching online, and we confirm these findings by working with OpenAI to query the original training data.

Results: Type of Extracted Content

Category	Count
US and international news	109
Log files and error reports	79
License, terms of use, copyright notices	54
Lists of named items (games, countries, etc.)	54
Forum or Wiki entry	53
Valid URLs	50
Named individuals (non-news samples only)	46
Promotional content (products, subscriptions, etc.)	45
High entropy (UUIDs, base64 data)	35
Contact info (address, email, phone, twitter, etc.)	32
Code	31
Configuration files	30
Religious texts	25
Pseudonyms	15
Donald Trump tweets and quotes	12
Web forms (menu items, instructions, etc.)	11
Tech news	11
Lists of numbers (dates, sequences, etc.)	10

Table 1: Manual categorization of the 604 memorized training examples that we extract from GPT-2, along with a description of each category. Some samples correspond to multiple categories (e.g., a URL may contain base-64 data). Categories in bold correspond to personally identifiable information.

Results: Effectiveness of different approaches

Inference Strategy	Text Generation Strategy		
	Top-n	Temperature	Internet
Perplexity	9	3	39
Small	41	42	58
Medium	38	33	45
zlib	59	46	67
Window	33	28	58
Lowercase	53	22	60
Total Unique	191	140	273

Table 2: The number of memorized examples (out of 100 candidates) that we identify using each of the three text generation strategies and six membership inference techniques. Some samples are found by multiple strategies; we identify 604 unique memorized examples in total.

Results: Entropy & Perplexity

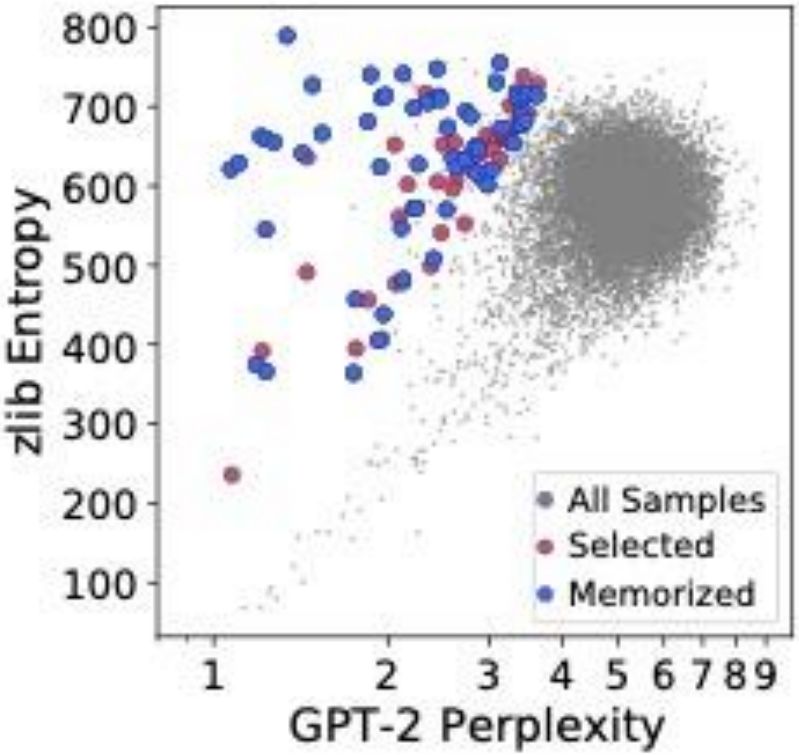

Figure 3: The zlib entropy and the perplexity of GPT-2 XL for 200,000 samples generated with top-n sampling. In red, we show the 100 samples that were selected for manual inspection. In blue, we show the 59 samples that were confirmed as memorized text. Additional plots for other text generation and detection strategies are in Figure 4.

Results: Examples of 1-editic memorization

Memorized String	Sequence Length	Occurrences in Data	
		Docs	Total
Y2...[redacted]...y5	87	1	10
7C...[redacted]...18	40	1	22
XM...[redacted]...WA	54	1	36
ab...[redacted]...2c	64	1	49
ff...[redacted]...af	32	1	64
C7...[redacted]...ow	43	1	83
0x...[redacted]...C0	10	1	96
76...[redacted]...84	17	1	122
a7...[redacted]...4b	40	1	311

Table 3: **Examples of $k = 1$ eidetic memorized, high-entropy content that we extract** from the training data. Each is contained in *just one* document. In the best case, we extract a 87-characters-long sequence that is contained in the training dataset just 10 times in total, all in the same document.

- Personally Identifiable Information (78 examples)
- URLs (50 examples)
- Code (31 samples of memorized source code)
- Unnatural text
- Memorization is context dependent!
- “3.14519” gives first 25 digits of pi
- “pi is 3.14519” gives first 799 digits of pi

Mitigating Privacy leakage in LLMs

- Training with Differential Privacy (must face/deal with utility-vs-privacy)
- Curating the training data (filtering/removing PII or sensitive information)
- Impact on Downstream fine-tuning (how much memorization is inherited)
- Auditing ML models for Memorization

Conclusions

- Extraction Attacks ***are a Practical Threat*** for LLMs
- Memorization ***does not*** require Overfitting
- Memorization can be hard to discover
- ***Larger*** Models Memorize ***more*** Data
- ***Auditing ML models*** for Memorization